Task 3

Problem 1

Not Boring Movies - LeetCode

```
SELECT *

FROM Cinema

WHERE  id %2!=0 and id != 3

ORDER BY id desc;
```

Problem 2

Project Employees I - LeetCode

```
SELECT P.project_id,ROUND(AVG(E.experience_years), 2) AS average_years

FROM Project AS P

INNER JOIN Employee AS E

ON P.employee_id = E.employee_id

GROUP BY P.project_id;
```

Problem 3

Monthly Transactions I - LeetCode

```
SELECT

    DATE_FORMAT(trans_date, '%Y-%m') AS month,
    country,

    COUNT(*) AS trans_count,

    SUM(state = 'approved') AS approved_count,

    SUM(amount) AS trans_total_amount,

    SUM(CASE WHEN state = 'approved' THEN amount ELSE 0 END) AS
approved_total_amount

FROM Transactions

GROUP BY

    DATE_FORMAT(trans_date, '%Y-%m'),country;
```

Problem 4

Number of Unique Subjects Taught by Each Teacher - LeetCode

```
SELECT teacher_id ,COUNT(DISTINCT subject_id) AS cnt

FROM Teacher

GROUP BY teacher_id ;
```

Problem 5

Employees Whose Manager Left the Company - LeetCode

```
SELECT employee_id

FROM Employees

WHERE salary < 30000 AND manager_id NOT IN (SELECT employee_id FROM
Employees);
```

Problem 6

Exchange Seats - LeetCode

```sql
SELECT

    CASE

        WHEN id % 2 = 0 THEN id - 1

        WHEN id % 2 = 1 AND id < (SELECT MAX(id) FROM seat)THEN id + 1

        else id

    END AS id,

    student

FROM Seat

ORDER BY id;
```

Problem 7

Primary Department for Each Employee - LeetCode

```sql
SELECT employee_id, department_id

FROM Employee

GROUP BY employee_id

HAVING count(employee_id)=1

UNION ALL

SELECT employee_id, department_id

FROM Employee
```

```sql
WHERE primary_flag='Y'
```

Part of Index problems (Problem 7)

```sql
CREATE NONCLUSTERED INDEX service_created
ON AppLogs (service_name, created_at DESC);
```

Problem 8

```sql
CREATE NONCLUSTERED INDEX customer_id
ON Orders(customer_id);
```

Problem 10

```sql
CREATE VIEW v_VIPCustomers AS
SELECT c.name,c.email,SUM(o.total_amount) AS total_spent
FROM Customers c
INNER JOIN Orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name, c.email
HAVING SUM(o.total_amount) > 5000
ORDER BY SUM(o.total_amount) DESC;
```